GACE: A meta-heuristic based in the hybridization of Genetic Algorithms and Cross Entropy methods for continuous optimization

P. Lopez-Garcia[a], E. Onieva[a], E. Osaba[a], A.D. Masegosa[a,b], A. Perallos[a]

[a] *Deusto Institute of Technology (DeustoTech), University of Deusto, Bilbao 48007, Spain.*
[b] *IKERBASQUE, Basque Foundation for Science, Bilbao 48011, Spain*

Abstract

Metaheuristics have proven to get a good performance solving difficult optimization problems in practice. Despite its success, metaheuristics still suffers from several problems that remains open as the variability of their performance depending on the problem or instance being solved. One of the approaches to deal with these problems is the hybridization of techniques. This paper presents a hybrid metaheuristic that combines a Genetic Algorithm (GA) with a Cross Entropy (CE) method to solve continuous optimization functions. The algorithm divides the population into two sub-populations, in order to apply GA in one sub-population and CE in the other. The proposed method is tested on 24 continuous benchmark functions, where each one is used in four different dimensions. First, a study to find the best parameter configuration is done. The best configuration found is compared with several algorithms in the literature in order to demonstrate the competitiveness of the proposal. The results shows that GACE is the best performing method for instances with high dimensionality. Statistical tests have been applied, to support the conclusions obtained in the experimentation.

Keywords: Genetic Algorithm, Cross Entropy, Hybrid Algorithm, Continuous Optimization, Optimization Functions

1. Introduction

In the last decades, metaheuristics have been used extensively to solve complex optimization problems. Many of these algorithms have been inspired by natural phenomena and have great value in solving high dimensional problems. In this category are algorithms like Particle Swarm Optimization (PSO)

Email addresses: p.lopez@deusto.es (P. Lopez-Garcia), enrique.onieva@deusto.es (E. Onieva), e.osaba@deusto.es (E. Osaba), ad.masegosa@deusto.es (A.D. Masegosa), perallos@deusto.es (A. Perallos)

(Kennedy, 2010), Genetic Algorithm (GA) (Holland, 1975), Ant Colony Optimization (ACO) (Dorigo & Gambardella, 1997), Differential Evolution (DE) (Neri & Tirronen, 2010), or Simulated Annealing (SA) (Van Laarhoven & Aarts, 1987). Since their formulation, these algorithms have been applied to optimization problems in (Wang et al., 2011; Thakur, 2014; Ciornei & Kyriakides, 2012; Cai & Ma, 2010). Also, probabilistic techniques such as Cross Entropy (CE) (Rubinstein, 1999) or Covariance Matrix Adaptation (CMA) (Hansen & Ostermeier, 2001) have been applied to this kind of problem (Kroese et al., 2006; Deb et al., 2002).

Despite its success in continuous problems and the large number of existing techniques, metaheuristics still suffers from several problems that remains open. One of them is the variability of its performance, depending on the characteristics of the optimization function. Another issue to take into account are the weaknesses strengths that each technique presents. For example, population-based metaheuristics like GA and ACO have problems with the exploitation of the search space (Talbi, 2002). On the other hand, regarding trajectory-based algorithms, as SA or Tabu Search, they easily become stuck in local optima because of their bad exploratory behaviour (Wang et al., 2004). In the case of DE, the specific way in which new individuals are created or the potential to generate only a limited number of different trial solutions within one generation are identified as problems to this method (Segura et al., 2015). One of the approaches to deal with these problems is the association of two or more algorithms in order to obtain a better one or counteract their drawbacks. In fact, choosing a satisfactory combination of algorithms can be an essential part for achieving better performance in many hard optimization problems. This combination is called Hybridization (Topcuoglu et al., 2007).

The principal aim to hybridize different algorithms is to benefit from the sinergy between their complementary weaknesses and strengths. Hybrid algorithms have proved to be promising in many fields in general (Purwar & Singh, 2015; Fujikawa & Takashi, 2005; Olama et al., 2015), and in particular in optimization problems such as constrained problems (Hernández et al., 2013), nonlinear problems (Abd-El-Wahed et al., 2011), or real world problems (Mandal et al., 2011; Asafuddoula et al., 2011).

In this article, a novel hybridization technique is presented. The proposal is based on a hybridization between GA and CE for solving continuous optimization problems. There are several reasons which have motivated the development of this study:

- The proposal aims at finding sinergies between the good exploration and exploitation abilities of GA and CE, respectively.
- Both methods have been successfully applied separately and many papers have been published focused on them (Wang et al., 2013; Zhao et al., 2013; Busoniu et al., 2011). However, as far as we know, the hybridization between these two methods has not been done before. Therefore, it could be an interesting approach to hybridize both method in order to achieve better performance than for its own.

The basic concept of the proposed technique is the following: the algorithm divides the population into two sub-populations of a given size. Then, GA is applied to one of these sub-populations and CE is applied to the other. As result, the new individuals created by the algorithms will form the new population. The algorithm has been tested over 24 benchmark functions extracted from Black-Box Optimization Benchmarking (BBOB)[1], which is part of the GECCO and CEC international conferences. Furthermore, it will be compared with reference algorithms in the literature to demonstrate its performance on this kind of problem. To conclude, the objectives of this paper can be summarized as follows.

- Hybridize two well-known algorithms, such as GA and CE, in order to improve on the performance obtained by those algorithms on their own.
- Apply this hybridization to continuous optimization problems.
- Find a successful combination of parameters to obtain a good performance in all the functions used.
- Compare the performance of the proposal with that of methods in the literature to prove its potential.

The work developed in this article is an extension of the research presented by the authors in Genetic and Evolutionary Computation Conference 2015 as two-page Late-Breaking Abstract (Lopez-Garcia et al., 2015). The novelties in this work are listed below:

- The number of functions used have been increased to 24, the double as in previous work.
- A wide study of the parameters used in the algorithm has been done. Population sizes and special parameters of each part of the method have been studied in order to obtain the best configuration possible.
- The number of different dimension values considered have been increased.
- The proposal have been compared with new high-performance methods from literature.
- Statistical tests have been applied in order to prove the significance of the results obtained by the presented method.

The proposal has been also used in real-world optimization in Lopez-Garcia et al.. In this paper, the proposal is focused in continuous optimization, and extend the parameter study part, which is basic for the good performance of the algorithm, not only in this kind of problems but for its use in other themes.

[1] http://coco.gforge.inria.fr/doku.php?id=bbob-2013

This article is structured as follows. In Section 2 a brief explanation of the different parts of the proposal and its application in the literature is given. A brief explanation about types of hybridized methods, and recent state of the art about them are described in Section 3. Section 4 explains how the proposal works and what operators are used. The results of the experimentation and an analysis of the results are presented in Section 5. Finally, some conclusions and avenues for further research are presented in Section 6.

2. Background

In this section, a brief background of the different parts of the proposal is given. In Section 2.1, an explanation of GA is given and some of the related literature is reviewed. On the other hand, Section 2.2 contains the description of CE and some related research.

2.1. Genetic Algorithm

GAs were introduced by Holland (1975).They were designed to mimic some of the processes observed in natural evolution. A GA maintains a population of solutions, called individuals, and iteratively modifies them using different operators in order to achieve improvements. Its adaptability to hard problems has led GAss to appear in the literature both on their own (Osaba et al., 2014, 2013), as well as combined with different techniques (Onieva et al., 2011; Qiao et al., 2011), to solve a wide variety of problems. A GA is formed most of time by the four operators: selection, crossover, mutation, and replacement.

The state of the art about GA is wide. Interested readers are referred to Lim (2014), Kumar & Beniwal (2013), and Karakatic & Podgorelec (2015) for extensive reviews of GAs in the literature.

In Algorithm 1, a pseudocode of the basic GA is depicted, where p_c and p_m denote the crossover and mutation probabilities, respectively.

Data: POP_{size}, p_c, p_m, T_{max}
Result: *Best individual found*
$t \leftarrow 0$
$POP_0 \leftarrow$ Initialize(POP_{size})
Evaluate POP_0
while $t < T_{max}$ **do**
  $Parents \leftarrow$ Select parents from POP_t
  $Offspring \leftarrow$ Crossover($Parents, p_c$)
  $Offspring \leftarrow$ Mutate($Offspring$,p_m)
  Evaluate $Offspring$
  $POP_{t+1} \leftarrow$ Replacement process with actual Population POP_t and $Offspring$
  $t \leftarrow t + 1$
end

Algorithm 1: Pseudocode of workflow followed by GA.

2.2. Cross Entropy

The CE method was proposed by Rubinstein in 1997 (Rubinstein, 1999). This adaptive method was created for rare-event probabilities and combinatorial optimization. CE has three main phases:

1. Generate POP_{size} random samples from a normal distribution with mean $\overline{m}$ and standard deviation s ($N(\overline{m}, s)$).
2. Select the n_{up} best samples from $Samples$.
3. Update $\overline{m}$ and s according to the n_{up} individuals with better fitness.

CE has been used in estimation or optimization problems and it can be applied to different fields. For example, in Caballero et al. (2015), the authors developed a multiobjective optimization CE procedure for combinatorial problems and tested it with multiobjective knapsack problems and multiobjective assignment problems. CE is used as an optimization method of basis functions for control policies in continuous-state discrete-action Markov decision processes in Busoniu et al. (2011).

In a standard execution of the method, $\overline{m}$ tends to locate itself over the point with the best results, while s become smaller, until both values are focused on the area of the best solutions found in the domain. The variation of these values is done with a parameter indicating the learning rate (L_r), whose values usually range within the interval $[0.6, 0.9]$ (Reale & OConnor, 2011). The update of means and standard deviation is done by combining the actual value of $\overline{m}$ and s with the mean and standard deviation of the n_{up} samples selected in an iteration (the best ones).

Algorithm 2 represents the process described before. The algorithm is presented for a one-dimensional problem; in the case of more dimensions, $\overline{m}$ and s must be vectors and each one of their dimensions should be treated separately.

Data: $POP_{size}, n_{up}, L_r, T_{max}$
Result: *Best individual found*
$\overline{m} \leftarrow$ Initialize Means
$s \leftarrow$ Initialize standard deviations
$t \leftarrow 0$
while $t < T_{max}$ **do**
  $POP_t \leftarrow$ Generate POP_{size} Samples under $\mathcal{N}(\overline{m}, s)$
  Evaluate POP_t
  $Samples \leftarrow$ Select the n_{up} best individuals from POP_t
  $\overline{m} \leftarrow UpdateMeans(L_r, \overline{m}, Samples)$
  $s \leftarrow UpdateDeviation(L_r, s, Samples)$
  $t \leftarrow t + 1$
end

Algorithm 2: Pseudocode of workflow followed by CE.

3. Hybridized methods

Hybridization is an important topic in the literature (Purwar & Singh, 2015; Hernández et al., 2013). In a hybridization, two or more techniques are combined in order to create synergies among them. The combined techniques have obtained good performance for resolving the specified problem, and the combination of them can improve the results obtained on their own.

Due to the increasing number of works in this area, not only for metaheuristics during the last two decades (Osman & Laporte, 1996; Talbi, 2009), but recently with its hybridization (Blum et al., 2011), some interesting articles about how they can be classified are found in the literature. For example, in Talbi (2002), a taxonomy about hybrid metaheuristics is presented. This paper also presents a large number of hybrid approaches classified according to that taxonomy. Another example, which continues with this idea, is presented in Raidl (2006). In this article, author combines the point of view of Cotta-Porras (1998) and one of the approaches presented in Alba (2005) by Blum et al. with the taxonomy named before. The most recent taxonomy can be found in Jourdan et al. (2009), where Talbi (2002) is extended in order to consider cooperative schemes between exact methods and metaheuristics. In order to clarify in which category the proposal is, an analysis of the different types of possible combination following the taxonomy given by Talbi (2002) is made. This classification is divided into design issues and implementation issues. For design issues, the options are:

- Low-level, where a given function of a metaheuristic is replaced by another metaheuristic, or High-level, where the different metaheuristics are self-contained.
- Relay, where a set of metaheuristics is applied one after another, or Teamwork, in which there are many parallel cooperating agents.
- Homogeneous, where all the combined algorithms use the same heuristic, or Heterogeneous, where different heuristics are used.
- Global, where all the algorithms search in the whole research space, or Partial, where each algorithm has its own search space.
- Specialist, that combines algorithms which solve different problems, or General, where all the algorithms solve the same target optimization problem.

While for implementation issues, it can be found:

- Specific, which only solve a small range of problems with much higher rates and lower cost, or General-purpose.
- Sequential, where algorithms work one by one, and Parallel, where each algorithm is working at the same time than the rest of them.

If the heuristics used for the hybridization works in a parallel way, they fall into one of three categories:

1. Static, if the number of tasks and the location of work are generated at compilation time.
2. Dynamic, if the number of task is fixed at compilation time, but the location of work is determined and/or changed at run-time.
3. Adaptive, where tasks can be created or deleted as a function of the load state of the parallel machine.

Focusing in design issues, the proposed algorithm in this work fall into the next categories:

- High-level: Both GA and CE are self-contained in the algorithm.
- Teamwork: Both parts of the algorithm work in parallel and cooperate at the end of its execution.
- Heterogeneous: GA is a population-based technique while CE is a statistical one.
- Partial: as it is mentioned with more detail in future sections, both algorithms are applied in two different sub-populations, which both are created in a different way.
- General: both parts are applied to the same optimization problem.

In the other hand, metaheuristics and their hybridizations have been widely used for optimization problems in the literature. In Kıran & Gündüz (2013) a hybridization of PSO and Artificial Bee Colony (ABC) is presented and applied to twelve basic numerical benchmark functions and energy demand estimation problems. PSO is also used in Chen et al. (2009) in combination with DE for global optimization of multimodal functions. Sierra et al. (2014) have combined a DE with a K-means algorithm for continuous optimization. The application of K-means helps to obtain more diversity in the population and skip local optimum values while a DE with its original operators is working. The results are compared with a DE and a PSO. An adaptive memetic technique combining a GA, DE, and an estimation of the distribution algorithm is developed in Shim et al. (2015). In Duan et al. (2013), a hybrid PSO with GA is proposed to solve the multi Unmanned Aerial Vehicle formation reconfiguration problem, which is processed as a parameter optimization problem. Authors applied a combination of ACO and GA to different classes of complex global continuous optimization problems in Ciornei & Kyriakides (2012). In Abadlia et al. (2014), authors present two different hybridization of PSO. In the first case, the technique is combined with a Tabu Search method, while in the second case, it is combined with a global search technique. The proposed hybridizations are used in fifteen diferent functions from the multi-objective literature. A hybrid Artificial Bee Colony method is developed in Bolaji et al. (2015). The algorithm is hybridized

in two parts: first, authors applied a Local Search technique as a refinement process within the employed operator of the original technique. The second phase uses a harmony search algorithm to replace the scout bee operator. The proposal is evaluated using a benchmark dataset composed by 12 problem instances. In Elsheikh (2014), continuous-time simulation optimization problems are solved applying derivate-based hybrid heuristics, with population-based metaheuristics among them. The proposed approach improves the solution quality in this kind of problems.

Hybridization is not only between metaheuristics. In Rahman et al. (2014), authors explore the concept of hybridizing metaheuristics with mixed integer programming solvers, and proposed a hybridization of PSO with a mixed integer programming solver. The literature about hybrid techniques is very wide. As can be seen, the most part of the previous techniques use a metaheuristic as base algorithm, and apply different kinds of search algorithms to improve its performance in a concrete point, or avoid problems like stagnation in local optima, loss of diversity, and so on. On the other way, there are also techniques that combine two techniques in order to alleviate the drawbacks of each other. The proposed technique in this work is focused on trying to improve the parts where the other one can obtain worse performance. More details about how the proposal works and its characteristics are explained in next sections.

Hybridization between GA and CE is not found in the literature, as far as we know. CE can improve the capacity of exploitation of a GA while GA can improve the capacity of exploration in a CE. The aim of this paper is to achieve the good performance of the combination of these techniques, and find the correct parameters for its application in continuous optimization problems.

4. Genetic Algorithm and Cross Entropy: GACE

The use of the proposed algorithms for this hybridization is justified as follow. As it has been mentioned in previous sections, one of the problems of population-based algorithms is their bad exploitation abilities. On the other side, CE method has a high possibility to become stuck in local optima if it is applied alone, since it tends to converge quickly to local optima, specially when learning rate is high. In the proposed technique, CE is used to cover the lack of exploitation of GA, focusing the search of solutions in the area where the best individuals have appeared. In this case, GA is responsible of the exploration of the search. Then, the GACE method is created with the aim of taking advantage of the exploration ability of a GA and the exploitation ability of a CE in continuous optimization. The algorithm works in this way: first, an initial population is created randomly with a given number of individuals POP_{size}. At each iteration of the algorithm, the population POP_t is divided into two sub-populations: POP_{GA} with $SIZE_{GA}$ individuals and POP_{CE} with $SIZE_{CE}$. $SIZE_{GA}$ is calculated using a percentage p_{ga} of POP_{size}, while $SIZE_{CE}$ is calculated as $POP_{size} - SIZE_{GA}$. The individuals in POP_{GA} are chosen by the selection method while the $SIZE_{CE}$ best individuals in the actual

population POP_t are selected to form POP_{CE}. Once the sub-populations are created, each one is used in a different way:

- POP_{GA}: GA operators are applied to this population in order to create $SIZE_{GA}$ new individuals. Selection, crossover, and mutation operators used are specified in Section 4.1.
- POP_{CE}: $SIZE_{CE}$ new individuals are randomly generated using a normal distribution with mean $\bar{M}$ and standard deviation S, $\mathcal{N}(\bar{M}, S)$, updated employing Algorithm 2. Section 4.2 presents in detail the implementation of this process.

After that, a population POP_{t+1} is created. It contains the individuals generated by the last two operations, i.e., POP_{t+1} will contain $SIZE_{GA}$ individuals generated by a GA and $SIZE_{CE}$ individuals created using a CE method. Then, the total population size is the sum of the number of individuals in each sub-population, i.e. $POP_{size} = SIZE_{GA} + SIZE_{CE}$. In addition, elitism is applied. In case the best individual found so far is not present in the generation, it is inserted, replacing the worst one. Algorithm 3 describes the whole process.

Data: $POP_{size}, p_{ga},\ p_c,\ p_m,\ L_r,\ p_{up},\ T_{max}$
Result: *Best individual found*
$SIZE_{GA} \leftarrow \parallel POP_{size} \cdot p_{ga} \parallel$
$SIZE_{CE} \leftarrow POP_{size} - SIZE_{GA}$
$n_{up} \leftarrow \parallel SIZE_{CE} \cdot p_{up} \parallel$
$t \leftarrow 0$
$POP_0, \leftarrow$ Initialize(POP_{size})
$\overline{M} \leftarrow$ Initialize Means vector
$S \leftarrow$ Initialize Standard Deviation vector
Evaluate POP_0
while $t < T_{max}$ **do**
    $POP_{GA} \leftarrow$ SelectionOperator($POP_t, SIZE_{GA}$)
    $POP_{CE} \leftarrow$ SelectBestSamples($POP_t, SIZE_{CE}$)
    $Offspring_{GA} \leftarrow$ Crossover(POP_{GA}, p_c)
    $Offspring_{GA} \leftarrow$ Mutation($Offspring_{GA}$,p_m)
    $Offspring_{CE} \leftarrow$ Generate($POP_{CE}, SIZE_{CE}, \overline{M}, S$)
    $\overline{M} \leftarrow UpdateMeans(L_r, \overline{M}, Offspring_{CE}, n_{up})$
    $S \leftarrow UpdateDeviation(L_r, S, Offspring_{CE}, n_{up})$
    $POP_{t+1} \leftarrow Offspring_{GA} \bigcup Offspring_{CE}$
    Evaluate POP_{t+1}
    Add the best individual found to POP_{t+1} if it is not in the population
    $t \leftarrow t + 1$
end

Algorithm 3: Pseudocode of the workflow followed by the proposed method GACE

As it has been explained, the individuals that form POP_{GA} contribute to help the exploration of the search space, while the individuals of POP_{CE} provide exploitation. When both sub-populations are joined into a new one (POP_t), individuals can be in both sub-populations in the next generation, helping with the improvement of the best solutions found.

4.1. Genetic Algorithm Operators

In this section, the operators used in the Genetic Algorithm part are explained. The selection, crossover and mutation operators are presented below.

As the selection operator, Binary Tournament is used (Goldberg & Deb, 1991). This operator chooses two random individuals and compares them. The winner of the tournament is the fittest individual. The process can be repeated as often as desired. In this case, $SIZE_{GA}$ individuals are taken.

BLX-α (Herrera et al., 1998) is adopted as the crossover operator. This operator takes two parents $A = (a_1, \ldots, a_i, \ldots, a_m)$ and $B = (b_1, \ldots, b_i, \ldots, b_m)$ for each element i and creates two offspring. These new individuals are generated using random values in the interval presented in Equation 1, with $\alpha \in [0, 1]$. In this work, α takes the value 0.5. With this value, the difference between the initial individual and the resulted one is not so big, but enough to have a change between them.

$$O_i = \mathcal{U}(min(a_i, b_i) - \alpha|a_i - b_i|,\ max(a_i, b_i) + \alpha|a_i - b_i|) \tag{1}$$

In this paper, Gaussian mutation (Bäck & Schwefel, 1993) is used as the mutation operator. Each element a_i of an individual is updated according to Equation 2.

$$a_i = \mathcal{N}(a_i, \frac{R_{max} - R_{min}}{10}) \tag{2}$$

Here, $\mathcal{N}$ is a normal distribution with mean a_i and the second parameter is standard deviation, which depends on the length $R_{max} - R_{min}$, where R_{max} and R_{min} are the upper and lower bounds of the individual.

4.2. Cross Entropy Operators

The individuals which are processed in the CE method are chosen in a deterministic way. The $SIZE_{CE}$ best individuals of POP constitute POP_{CE}. For the good performance of CE method, two registers must be kept by CE. One of them stores the average mean $\overline{M} = (\overline{m}_1, \overline{m}_2 \ldots \overline{m}_n)$ and the other stores the standard deviation $S = (s_1, s_2, \ldots s_n)$. For each position i in the vectors, a random number is generated following a normal distribution $\mathcal{N}(m_i, s_i)$. Then, once the POP_{CE} individuals are chosen, the individuals $\overline{M}$ and S are updated and $SIZE_{CE}$ new samples are generated.

The update is done by applying Equations 3 and 4. The value of learning rate L_r will be studied in the next section.

$$\overline{M} = (1 - L_r) \cdot \overline{M} + L_r \cdot \overline{Samples} \tag{3}$$

Functions notation	Type
F_1, F_2, F_3, F_4, F_5	Separable
F_6, F_7, F_8, F_9	Low or Moderate conditioning
$F_{10}, F_{11}, F_{12}, F_{13}, F_{14}$	High conditioning and unimodal
$F_{15}, F_{16}, F_{17}, F_{18}, F_{19}$	Multi-modal with adequate global structure
$F_{20}, F_{21}, F_{22}, F_{23}, F_{24}$	Multi-modal with weak global structure

Table 1: Function notation and its type

$$S = (1 - L_r) \cdot S + L_r \cdot \sigma(Samples) \tag{4}$$

For their initialization, each component of $\overline{M}$ is initialized to a random value in the interval of the corresponding variable ($[R_{min}, R_{max}]$). On the other hand, S is equal to the amplitude of the interval ($R_{max} - R_{min}$).

5. Experimentation

In this section, the results of the experimentation employing the proposed algorithm are presented. In Section 5.1, the functions used and a brief explanation about its characteristics will be made. The study of the different parameters used in the application of GACE is developed in Section 5.2 while a comparison among the best possible configurations and techniques in the literature will be given in Section 5.3.

5.1. Continuous Optimization Functions

A total of 24 optimization functions from Comparing Continuous Optimisers (COCO) has been used for the present experimentation. The COCO platform has been used for the Black Box Optimization Benchmarking (BBOB) workshops that took place during the GECCO and CEC conferences from 2009 on. The functions used are noise-free and are of different types. The tables in Appendix A contain the name of each function, and its formula. Besides, Table 1 shows a summary of the functions that are contained in each type.

The reason to use these functions is to assess the performance of the proposed method for different kinds of functions, where some of them are known to be very complex. The constants and values of the functions can be found on the COCO webpage[2]. These functions have been optimized between $R_{min} = -5$ and $R_{max} = 5$ as it is indicated in the documentation provided by the BBOB competition. Besides, the dimensions in which functions used are optimized take values $dim \in \{5, 10, 20, 40\}$. To avoid these algorithms' always converging to the same value, a total of 15 instances with different optimum values were created for each of these functions.

[2]http://coco.gforge.inria.fr/doku.php

Parameter	Values
POP_{size}	$\{2, 5, 10, 20\} \cdot dim$
Percentage of GA individuals	$p_{ga} = \{0.25, 0.5, 0.75\}$
GA part parameters	$p_c = 0.9$, $p_m = \frac{1}{dim}$, $\alpha = 0.5$
CE part parameters	$L_r = 0.7$, $p_{up} = 1$
Stop Condition	$T_{max} = 25000$

Table 2: Values of the different parameters used in experimentation

	$c = 2$			$c = 5$			$c = 10$			$c = 20$		
p_{ga}	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
$Rank_{dim=5}$	8.38	8.33	6.33	5.25	**4.50**	5.83	6.13	5.50	**5.08**	8.29	6.25	5.13
$Rank_{dim=10}$	5.17	5.58	5.63	**4.50**	4.75	**4.54**	7.38	6.29	6.29	11.00	8.79	8.04
$Rank_{dim=20}$	**2.63**	**3.29**	3.92	5.17	4.42	5.46	8.63	6.58	6.88	11.67	9.88	9.50
$Rank_{dim=40}$	3.25	**2.67**	**2.75**	5.25	4.33	5.38	9.00	7.33	7.54	11.25	9.83	9.42
$Rank_{all}$	5.00	4.75	5.00	3.75	**3.00**	5.00	8.50	6.50	6.25	11.50	10.25	8.25

Table 3: Ranking of the different algorithms used by dimension and p_{ga}

5.2. GACE parameter tuning

In this section, the tuning of the parameters used by the proposed method is detailed. First, the values of these parameters was specified. In order to set the population size for future research and experimentation, it modifies its value depending on the value given to the dimension. The calculation of the population size POP_{size} is shown in Equation 5.

$$POP_{size} = c \cdot dim, \quad c \in \{2, 5, 10, 20\}, dim \in \{5, 10, 20, 40\} \tag{5}$$

i.e. $POP_{size} \in [10, 800]$.

For each one of the possible values POP_{size} can take, different percentages of the population associated to POP_{GA} (p_{ga}) are assumed. These percentages take 25%, 50%, and 75% of the population size, i.e. $p_{ga} = \{0.25, 0.5, 0.75\}$. Then, $SIZE_{GA}$ varies in the interval $SIZE_{GA} \in [3, 600]$ and $SIZE_{CE} \in [2, 600]$. For example, with $dim = 5$ and $c = 2$, $SIZE_{GA} \in \{3, 5, 8\}$ and $SIZE_{CE} \in \{7, 5, 2\}$. The proposal uses as its stop condition 25000 evaluations. Table 2 summarizes the default values of each of the parameters used in this experimentation.

First of all, a study to determine the value of c and a first look at p_{ga} was made. In order to show the results obtained in a simpler way, Table 3 provides the ranks obtained by dimension used. Each value represents the average position each combination of values obtains taking into account the error they have in each dimension and function used. The smaller the value, the better.

In this way, the authors want to find out which population size and which percentage of $SIZE_{GA}$ is needed to obtain the best results for the most functions. Bold values represent the best two rankings in each of the dimensions considered.

For $dim = 5$ functions, the best values were obtained in configurations with $c = \{5, 10\}$ and $p_{ga} = \{0.5, 0.75\}$. For $dim = 10$, the three best values are in the $c = 5$ configurations. For functions with $dim = \{20, 40\}$, $c = 2$ with all their

possible p_{ga} values include all the best rankings so far. Based on the results obtained, it can be said that for dimensions less than or equal to $dim = 10$, $c = 5$ can be chosen as the constant. In other cases, the constant value taken is $c = 2$. In a formal way, the formula for POP_{size} can be defined as in Equation 6:

$$POP_{size} = \begin{cases} 5 \cdot dim, \text{ if } dim \leq 10 \\ 2 \cdot dim, \text{ otherwise} \end{cases} \quad (6)$$

Given these values, a precise value for the percentage of $SIZE_{GA}$ is necessary. For this purpose, and following the previous formula, a study of the values of p_{ga} was made. To be exhaustive, the values $p_{ga} \in \{0, 0.05, 0.1, \ldots 1\}$ were taken. Besides, an important parameter for the CE part is included in this study. The percentage p_{up} needs to be defined in order to know how many individuals in POP_{CE} are used to update the means and standard deviation, i.e. p_{up} is needed to determine n_{up}. Hence, for this study, this value is defined as $p_{up} \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Each pair (p_{ga}, p_{up}) has been studied in order to determine the best performing combination. For this purpose, Figure 1 shows the different heat maps denoting the ranking obtained by the combination of parameters in each dimension. The darker the box, the better the ranking. Table 4 summarizes all the parameters used for (p_{ga}, p_{up}) experimentation.

Parameter	Values
POP_{size}	Equation 6
Percentage of GA individuals	$p_{ga} = \{0, 0.05, 0.1 \ldots 1\}$
Number of individuals for update	$p_{up} = \{0.2, 0.4 \ldots 1\}$
Stop Condition	$T_{max} = 25000$

Table 4: Values of the different parameters used in $p_{ga} - p_{up}$ experimentation

Along the X-axis, the percentages of $SIZE_{GA}$ are represented, while along the Y axis, the p_{up} values are shown. The upper figures show the results for dimensions 5 and 10 while the lower correspond to those with dimensions 20 and 40. A legend with the ranking values obtained in each square is shown in the right part of each figure. For $dim = 5$, the best values are found in the intervals $p_{up} \in [0.6, 1]$ and $p_{ga} \in [0.3, 0.7]$, while the worst values are situated in the $p_{up} = 0.2$ row and the $p_{ga} = 0$ column. For the remaining dimensions, the best results are concentrated in $p_{up} \in [0.2, 0.6]$ and $p_{ga} \in [0, 0.3]$. When the dimensions increase, it is easy to see that the best combinations are obtained with a low value of both parameters. The bigger the dimension, the larger the difference between the best and worst rankings.

One of the combinations that obtains good performance in higher dimensions is $p_{up} = 0.4$ and $p_{ga} = 0.1$. These values are used in the comparison with the existing techniques in the literature in the next section.

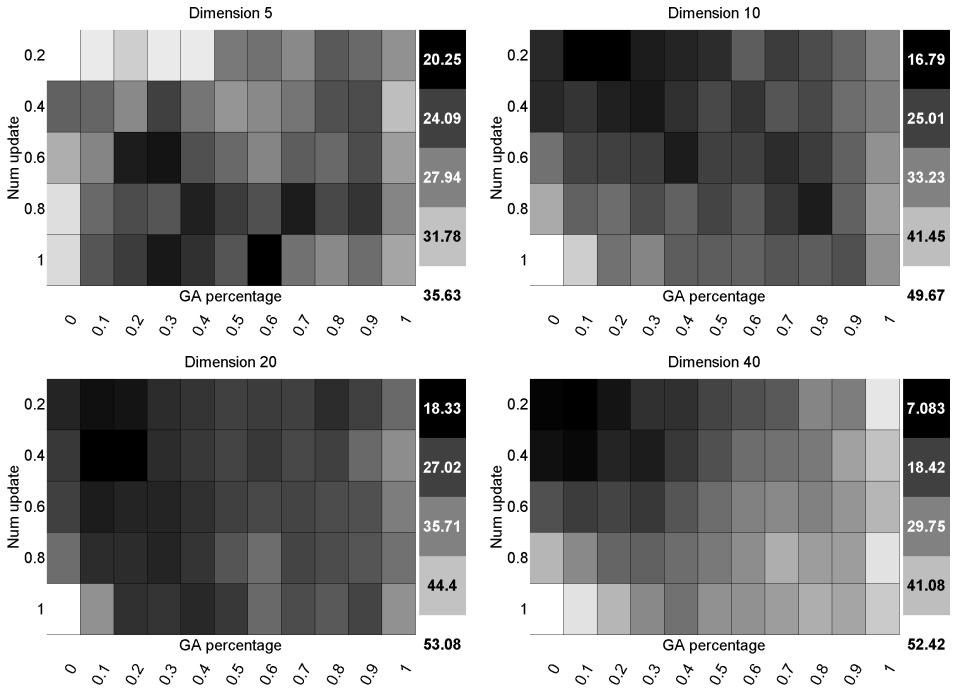

Figure 1: Ranking of each configuration used in different dimensions

5.3. Comparison with existing techniques from the literature

This section shows the comparison of the proposal employing the parameters chosen in previous section, with standard existing techniques in the literature. Standard techniques belonging to BBOB 2013 have been taken in order to provide a qualitative comparison with the proposal. These techniques include a real-coded genetic algorithm, two types of cellular genetic algorithm, a generational GA, a Differential Evolution algorithm, and a hill climber. Other techniques such as different variations of CMA algorithm Auger et al. (2013); Hutter et al. (2013), memetic algorithms Voglis (2013) or variations of NSGA-II Tran et al. (2013) have been also applied to this benchmark in 2013 edition.

The reason to make a comparison with these techniques is that they have been achieved a good performance on the same optimization functions. Besides, the comparison with several types of techniques in the literature such as a really powerful GA (the case of the real-coded one), a technique with the same approach as it (in the case of the cellular GA) in all its variations, a well-known technique as DE, or a simple technique, such as hill-climber, could provide a good and rich comparison with the proposal, in order to show the competitiveness of its performance.

The configurations used in the application of these techniques was that indicated by their authors in their original papers. The characteristics of the techniques are presented in what follows.

- Real-coded genetic algorithm based on enhanced projection (PRCGA). This algorithm uses five operators: tournament selection, blend-α crossover, non-uniform mutation, projection, and a stagnation alleviation mechanism. The crossover probability has been set to 0.8, the mutation probability to 0.15, and the tournament size to 3. The experimental procedure, conclusions, and more details can be consulted in (Sawyerr et al., 2013).

- Differential Evolution (DE) applied in Pošík & Klemš (2012) in its standard form, with 'best/1' mutation operator.

- Cellular Genetic algorithms are the most fine-grained of the Parallel Genetic algorithms. CGAs are one sort of GAs, in which the population is divided into semi-isolated subpopulations. In this case, each individual is its own subpopulation. Two of these algorithms are used for comparison. One of them, called *Grid*, is implemented in its standard form as described in (Alba & Dorronsoro (2009)). Also used is a one-directional ring CGA, *Ring*, developed in (Holtschulte & Moses, 2013).

- A generational, single-population genetic algorithm (*GGA*) using rank selection is used for comparison.

- A hill climber algorithm (*Hill*) (Jacobson, 2004) is an iterative optimization technique which belongs to the family of local search techniques. In this comparison, the technique developed in Holtschulte & Moses (2013) is used.

Tables 5–8 show the average error, where $error = F_{best} - F_{opt}$, for each dimension and function, where F_{best} is the best value obtained by the algorithm and F_{opt} is the optimal value of the instance used. As stop conditions, the same number of evaluations used in the proposed technique are employed in the baseline techniques. Bold values represent the two best values obtained for each function. Results with $*$ indicate the best error value reached in each function.

For $dim = 5$ functions, the proposal obtains one of the best two values in 11 of 24 functions, and the best so far in 3 of them. The best result is obtained by *DE* with 16 of 24 best values, and the best so far in 14 of them. *PRCGA*, *Ring*, *GGA* and show a similar performance with 5, 7 and 7 results out of 24, respectively. The techniques with fewer best values are *Grid*, which only obtains the best value in two cases, and *Hill*, which obtains the best results in four functions. The best results obtained by the proposed technique in this dimension are obtained mostly in Separable and Multi-modal with adequate global structure functions, and two of the most difficult high conditioning and unimodal functions. On the other hand, *DE* obtains its best values in Low or moderate conditioning and High conditioning and unimodal functions.

In the case of the experimentation with $dim = 10$, *Ring* and *Grid* have fewer good results: only one and four respectively, being the worst techniques in this dimension. For *DE*, it obtains one of the two best values in 13 out of 24, while the best value so far is obtained in 8 of these cases. However,

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**0.00e+00***	**7.08e-09**	1.19e-04	1.03e-04	1.58e-05	2.28e-04	2.70e-05
F_2	**6.62e-06**	**6.13e-09***	1.70e+00	3.35e+00	2.09e+00	1.35e+00	1.81e-03
F_3	**2.72e-03**	2.00e-01	5.44e-02	7.20e-02	**1.49e-02***	1.24e-01	4.15e-01
F_4	1.93e+00	1.89e+00	**9.50e-02**	2.15e-01	**3.99e-02***	2.46e-01	8.67e-01
F_5	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**2.24e+00**
F_6	2.48e-01	**7.58e-09***	1.53e-01	4.44e-01	**9.16e-02**	2.80e-01	1.60e+00
F_7	6.12e-01	**4.49e-09***	**1.38e-01**	6.98e-01	9.74e-01	2.04e-01	2.26e-01
F_8	3.54e+00	**5.29e-01***	1.47e+00	1.69e+00	1.40e+00	**8.25e-01**	1.50e+00
F_9	3.85e+00	**5.30e-01***	2.39e+00	3.78e+00	3.47e+00	1.76e+00	**1.73e+00**
F_{10}	5.04e+02	**2.61e-02***	2.88e+02	9.87e+02	8.43e+02	**2.09e+02**	2.06e+03
F_{11}	2.37e+01	**1.38e-03***	**6.42e+00**	3.54e+01	1.38e+01	8.07e+00	1.99e+01
F_{12}	7.05e+00	**1.71e+00***	7.48e+01	6.38e+01	5.73e+01	3.82e+02	**5.78e+00**
F_{13}	**3.76e+00**	**1.34e-03***	1.34e+01	1.15e+01	3.12e+01	8.34e+00	4.59e+00
F_{14}	**5.05e-04**	**9.94e-08***	3.14e-03	1.00e-02	9.80e-03	5.81e-03	2.90e-03
F_{15}	**3.11e+00**	**2.03e+00***	3.61e+00	1.12e+01	1.41e+01	5.78e+00	6.96e+00
F_{16}	**3.14e-01***	9.87e-01	**4.66e-01**	1.01e+00	1.59e+00	6.51e-01	5.98e-01
F_{17}	**6.24e-03**	**1.38e-04***	6.15e-02	4.27e-01	1.01e+00	1.40e-01	8.71e-02
F_{18}	**1.18e-01**	**2.24e-02***	3.31e-01	2.44e+00	2.43e+00	6.10e-01	2.72e-01
F_{19}	**1.77e-01**	4.01e-01	3.83e-01	6.96e-01	6.36e-01	6.14e-01	**3.04e-02***
F_{20}	5.63e-01	2.48e-01	**1.94e-01**	4.80e-01	5.04e-01	**1.68e-01***	8.66e-01
F_{21}	2.35e+00	8.38e-01	6.50e-01	1.99e+00	2.80e+00	**6.06e-04***	**4.73e-01**
F_{22}	1.62e+00	1.09e+00	**1.04e+00**	2.78e+00	1.59e+00	**1.49e-01***	2.32e+00
F_{23}	9.93e-01	1.19e+00	1.00e+00	**8.84e-01***	9.60e-01	**9.06e-01**	9.33e-01
F_{24}	9.81e+00	**9.76e+00**	1.11e+01	1.21e+01	1.08e+01	1.21e+01	**7.71e+00***

Table 5: Average $F_{best} - F_{opt}$ of techniques for $dim = 5$

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**0.00e+00***	**8.36e-09**	1.82e-03	1.42e-03	1.68e-04	1.05e-02	4.21e-06
F_2	**1.53e-05**	**8.27e-09***	1.36e+01	1.65e+01	1.30e+01	4.27e+01	1.07e-01
F_3	3.02e+00	1.12e+00	**6.38e-01**	1.09e+00	**8.14e-02***	2.54e+00	1.54e+00
F_4	7.88e+00	3.08e+00	**1.27e+00**	1.74e+00	**2.65e-01***	4.08e+00	2.47e+00
F_5	**5.95e-13**	**5.86e-15***	**5.86e-15***	**5.86e-15***	**5.86e-15***	**5.86e-15***	9.52e+00
F_6	3.10e+00	**1.06e-02***	**1.36e+00**	5.17e+00	2.49e+00	4.48e+00	1.76e+00
F_7	**2.18e+00**	**1.82e-01***	2.33e+00	7.08e+00	6.11e+00	3.23e+00	4.08e+00
F_8	9.58e+00	**2.38e+00**	1.48e+01	2.23e+01	**2.13e+00***	1.02e+01	1.12e+01
F_9	2.52e+01	**5.53e+00**	1.15e+01	7.33e+01	5.55e+01	1.27e+01	**4.76e+00***
F_{10}	**3.55e+03***	1.31e+04	**4.47e+03**	1.81e+04	1.84e+04	6.61e+03	6.40e+03
F_{11}	4.64e+01	8.39e+01	**3.15e+01***	9.79e+01	6.87e+01	4.26e+01	**4.00e+01**
F_{12}	**3.13e+00***	1.91e+02	1.28e+03	2.17e+03	1.33e+03	1.17e+04	**1.71e+01**
F_{13}	**6.17e+00**	**2.38e+00***	1.73e+01	3.65e+01	1.80e+01	3.93e+01	1.15e+01
F_{14}	**3.60e-03**	**7.49e-04***	1.61e-02	2.70e-02	1.44e-02	3.24e-02	4.54e-03
F_{15}	**6.79e+00***	3.85e+01	2.27e+01	5.93e+01	7.61e+01	3.87e+01	**1.28e+01**
F_{16}	**1.44e+00***	1.01e+01	**2.40e+00**	6.39e+00	6.28e+00	3.41e+00	2.43e+00
F_{17}	**1.50e-02**	**2.98e-03***	2.89e-01	3.11e+00	3.25e+00	9.11e-01	2.75e-01
F_{18}	**1.82e-01***	**4.22e-01**	1.32e+00	1.06e+01	1.61e+01	3.11e+00	1.09e+00
F_{19}	**1.27e+00**	2.81e+00	2.37e+00	3.46e+00	3.21e+00	3.00e+00	**3.30e-02***
F_{20}	1.29e+00	**3.55e-01***	**5.16e-01**	7.17e-01	6.65e-01	7.13e-01	1.30e+00
F_{21}	3.74e+00	2.77e+00	**1.92e+00**	6.03e+00	1.48e+01	**4.36e-01***	4.01e+00
F_{22}	7.80e+00	**2.21e+00**	4.22e+00	1.47e+01	6.59e+00	**7.51e-01***	4.79e+00
F_{23}	1.66e+00	1.90e+00	1.41e+00	1.47e+00	**1.32e+00***	**1.38e+00**	1.45e+00
F_{24}	**3.56e+01**	5.03e+01	4.67e+01	6.39e+01	5.59e+01	5.56e+01	**2.93e+01***

Table 6: Average error of techniques for $dim = 10$

PRCGA improves its performance to 6 out of 24 best values. The proposal in this case improves as well, obtaining one of the best results in 14 out of 24 functions, and the best so far in 6, being the second best technique for this value of the dimension. Focusing in the best techniques in this dimension, *DE* and the proposal, the first one obtains the most of its best values in Low and moderate type functions, while the proposal gets again the best results in Multi-modal with adequate global structure functions as well as in 4 of the 5 High conditioning and unimodal functions.

For experimentation with functions with $dim = 20$, the best techniques are *GACE* and *PRCGA*, with, respectively, 18 and 9 out of 24. In this case, *GACE*

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**5.13e-05**	**8.62e-06***	2.65e-02	2.12e-02	1.33e-03	2.56e-01	1.82e-04
F_2	**1.13e-02***	**3.94e-02**	1.10e+02	2.23e+02	6.54e+01	8.76e+02	3.16e+01
F_3	9.87e+00	9.45e+01	**5.71e+00**	7.61e+00	**8.55e-01***	2.33e+01	6.20e+00
F_4	2.63e+01	9.07e+01	**9.90e+00**	1.09e+01	**1.53e+00***	3.34e+01	1.20e+01
F_5	2.85e-09	**-3.61e-15***	**6.46e-14**	**6.46e-14**	**6.46e-14**	**6.46e-14**	1.19e+01
F_6	**1.47e+01***	1.80e+02	**1.49e+01**	4.60e+01	2.66e+01	6.66e+01	2.57e+01
F_7	**1.32e+01**	2.13e+01	**1.22e+01***	4.14e+01	3.38e+01	2.79e+01	1.72e+01
F_8	**2.64e+01**	**2.59e+01***	7.72e+01	9.49e+01	4.80e+01	8.26e+01	6.46e+01
F_9	**3.66e+01**	**2.81e+01***	5.29e+01	9.44e+01	7.62e+01	1.01e+02	4.03e+01
F_{10}	**3.54e+04**	3.27e+05	4.26e+04	8.23e+04	4.28e+04	4.68e+04	**3.11e+04***
F_{11}	**8.86e+01***	2.46e+02	9.76e+01	2.04e+02	2.21e+02	1.51e+02	**9.04e+01**
F_{12}	**3.13e+02**	3.89e+02	2.46e+04	1.90e+04	1.42e+03	2.60e+05	**1.10e+02***
F_{13}	**2.16e+01**	**1.27e+01***	6.67e+01	6.10e+01	2.46e+01	1.49e+02	2.19e+01
F_{14}	**1.31e-02***	2.33e-02	5.25e-02	6.40e-02	2.65e-02	3.05e-01	**1.88e-02**
F_{15}	**2.11e+01***	1.47e+02	9.16e+01	2.29e+02	2.61e+02	1.48e+02	**4.26e+01**
F_{16}	**2.53e+00***	2.88e+01	7.73e+00	1.36e+01	1.35e+01	1.09e+01	**5.45e+00**
F_{17}	**1.75e-01***	**5.70e-01**	6.69e-01	8.13e+00	1.12e+01	3.29e+00	7.40e-01
F_{18}	**8.47e-01***	5.39e+00	4.76e+00	2.91e+01	4.24e+01	1.20e+01	**2.70e+00**
F_{19}	**2.90e+00**	6.21e+00	5.24e+00	7.78e+00	7.35e+00	6.74e+00	**2.06e-01***
F_{20}	2.29e+00	2.66e+00	**9.07e-01**	1.17e+00	**8.82e-01***	1.44e+00	1.63e+00
F_{21}	**4.61e+00**	5.73e+00	5.01e+00	1.20e+01	1.86e+01	**1.54e+00***	1.12e+01
F_{22}	9.15e+00	8.80e+00	**7.55e+00**	2.38e+01	1.31e+01	**3.33e+00***	7.93e+00
F_{23}	2.43e+00	3.45e+00	2.66e+00	**2.42e+00**	**2.10e+00***	2.51e+00	2.42e+00
F_{24}	**1.45e+02**	1.60e+02	1.51e+02	2.38e+02	2.51e+02	2.07e+02	**1.11e+02***

Table 7: Average error of techniques for $dim = 20$

obtains 14 out of 18 of its best values in Low or moderate conditioning, High conditioning and unimodal, and Multi-modal with adequate global structure type functions, while *PRCGA* obtains 8 of its 9 best values in High conditioning and unimodal, and Multi-modal with adequate global structure type functions. The difference between *GACE* and *DE* in terms of the number of their good results has increased over the previous dimension. For this value of dimension, *DE* obtains only 7 out of 24 best values, where 5 of them are the best so far. In the case of *Grid* and *Ring*, they still obtain the worst results. Also, the proposal reaches the best value so far in 8 functions, and one of the two best values in at least one function of each type.

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**3.98e-08***	7.33e+00	4.25e-01	2.62e-01	1.17e-02	3.93e+00	**7.27e-03**
F_2	**8.52e+00***	3.12e+04	2.26e+03	2.71e+03	6.74e+02	2.35e+04	**8.04e+01**
F_3	4.31e+01	3.98e+02	4.21e+01	**3.99e+01**	**7.13e+00***	1.43e+02	4.51e+01
F_4	6.65e+01	6.83e+02	**5.77e+01**	6.27e+01	**1.12e+01***	2.05e+02	7.83e+01
F_5	**2.46e-01**	**1.34e-14***	**1.34e-14***	**1.34e-14***	**1.34e-14***	**1.34e-14***	3.99e+01
F_6	**6.77e+01***	3.44e+03	1.35e+02	4.42e+02	1.56e+02	3.95e+02	**1.06e+02**
F_7	**3.97e+01***	4.92e+02	9.57e+01	1.70e+02	1.57e+02	1.56e+02	**6.61e+01**
F_8	**5.44e+01***	3.39e+03	2.59e+02	2.45e+02	**1.06e+02**	8.31e+02	1.82e+02
F_9	**4.28e+01***	2.88e+03	2.48e+02	4.33e+02	**9.38e+01**	1.04e+03	1.89e+02
F_{10}	**1.12e+05***	2.94e+06	1.76e+05	3.73e+05	1.54e+05	4.18e+05	**1.24e+05**
F_{11}	**1.90e+02***	7.06e+02	2.91e+02	4.86e+02	4.56e+02	3.51e+02	**1.99e+02**
F_{12}	**5.44e+02***	2.44e+07	4.22e+05	3.36e+05	3.29e+04	5.40e+06	**8.63e+03**
F_{13}	**7.06e+01**	9.69e+02	2.25e+02	1.90e+02	**5.79e+01***	5.89e+02	9.96e+01
F_{14}	**2.65e-02***	1.91e+01	3.40e-01	2.54e-01	**5.57e-02**	4.02e+00	8.27e-02
F_{15}	**8.99e+01***	6.04e+02	2.96e+02	7.58e+02	8.44e+02	5.81e+02	**1.47e+02**
F_{16}	**1.05e+01**	4.48e+01	1.52e+01	2.67e+01	2.40e+01	2.29e+01	**9.34e+00***
F_{17}	**1.99e-01**	7.75e+00	1.79e+00	1.34e+01	2.11e+01	7.84e+00	**1.49e+00***
F_{18}	**1.01e+00***	3.01e+01	8.69e+00	5.14e+01	7.92e+01	2.73e+01	**5.98e+00**
F_{19}	**5.90e+00**	9.50e+00	8.02e+00	1.29e+01	1.16e+01	9.68e+00	**2.50e-01***
F_{20}	3.14e+00	8.55e+01	1.57e+00	**1.52e+00**	**8.77e-01***	2.89e+00	1.81e+00
F_{21}	5.98e+00	4.82e+01	6.35e+00	7.15e+00	1.69e+01	**5.27e+00**	**4.95e+00***
F_{22}	**7.83e+00**	3.91e+01	9.00e+00	2.30e+01	1.74e+01	**5.43e+00***	9.49e+00
F_{23}	4.06e+00	5.97e+00	4.24e+00	**3.33e+00**	**2.99e+00***	3.66e+00	4.39e+00
F_{24}	**3.87e+02**	6.36e+02	**3.80e+02***	7.68e+02	8.19e+02	6.18e+02	4.42e+02

Table 8: Average error of techniques for $dim = 40$

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
$Rank_{dim=5}$	3.69	**2.44**	3.60	5.5	4.81	3.83	4.12
$Rank_{dim=10}$	3.19	**3.02**	3.23	5.85	4.60	4.69	3.42
$Rank_{dim=20}$	**2.46**	4.21	3.5	5.35	4.46	5.12	2.89
$Rank_{dim=40}$	**2.08**	6.33	3.54	4.71	3.75	4.71	2.87
$Rank_{all}$	**2.85**	4	3.47	5.35	4.41	4.59	3.33

Table 9: Friedman test results for each dimension

Finally, for $dim = 40$ functions, the techniques that obtain the worse values are *Ring* and *GGA*, each with 3 out of 24, and *DE*, with only one best value. *Grid* obtains four best values while *Hill* obtains better results than it did for $dim = 20$, with 9 of 24 functions. This time, *GACE* is the best technique, with one of the two best values in 19 out of 24 functions (12 best values so far), followed by *PRCGA* with 13 bold values. The most part of these values are obtained, in the case of the proposal, in the same type functions than in the previous dimension. The same occurs for *PRCGA* technique.

To sum up, in a total of 96 studied cases, *GACE* obtains one of the two best values in 62 cases, while it reaches the best value so far in 29. The second best technique is *DE* with one result between the best ones in 37 of 96 cases, and with the best so far in 28.

To assess whether the differences in performance observed in the previous tables are significant or not, the use of statistical tests is necessary. Two statistical tests have been performed, following the guidelines proposed in Derrac et al. (2011).

First of all, the Friedman test (Derrac et al., 2011) has been applied to check whether there are significant differences between the seven methods compared. Table 9 shows the mean ranking provided by this non-parametric test for each algorithm in each dimension, and globally over all dimensions. The smaller the rank is, the better.

The proposal obtains the best ranking in $dim = 20$ and $dim = 40$, and in terms of $Rank_{all}$, where all dimensions are taking into account. The difference between the proposal's ranking and the different algorithms is bigger when the dimensions value is high, and, therefore, the complexity of the problem, is increased.

Holm's (Holm, 1979) and Finner's (Finner, 1993) post-hoc tests have also been applied to the results obtained by the Friedman procedure using *DE* as control method in $dim = \{5, 10\}$, and the proposal in the rest of dimensions. Table 10 shows the adjusted p-values returned by the tests of Holm and Finner for each dimension and technique. These values are rounded to three decimals. In order to highlight the significant differences, those p-values lower than 0.05 are in bold.

Globally, *GACE* is significantly better than all the methods except one, PRCGA. In this case, although the difference are not significant, the confidence level is still high, concretely, 88%. Looking at the results in each dimension,

	$dim = 5$		$dim = 10$			$dim = 20$		$dim = 40$		Global	
	Holm	Finner	Holm	Finner		Holm	Finner	Holm	Finner	Holm	Finner
GACE	0.09	0.054	1.577	0.799	DE	**0.015**	**0.007**	**0**	**0**	**0.001**	**0**
GGA	0.09	0.061	1.577	0.799	GGA	0.189	0.113	**0.039**	**0.023**	**0.097**	**0.058**
Grid	**0**	**0**	**0**	**0**	Grid	**0**	**0**	**0**	**0**	**0**	**0**
Hill	**0.001**	**0.001**	0.062	**0.022**	Hill	**0.005**	**0.003**	**0.022**	**0.011**	**0**	**0**
Ring	0.076	**0.037**	**0.038**	**0.022**	Ring	**0**	**0**	**0**	**0**	**0**	**0**
PRCGA	**0.027**	**0.013**	1.577	0.673	PRCGA	0.483	0.483	0.204	0.204	0.128	0.128

Table 10: Holm and Finner statistics for each dimension and technique

when $dim = 5$, DE improves significantly to $Grid$, $Hill$ and $PRCGA$. For $dim = 10$, Finner's test indicates that DE improves significantly $Grid$, $Hill$ and $Ring$ methods, while, according to Holm, the proposal only improves $Grid$ and $Ring$. Finally, for $dim = \{20, 40\}$, both tests indicate that $GACE$ improves significantly all the methods in $dim = 20$, except $PRCGA$, where, for $dim = 40$, the values of $GACE$ and $PRCGA$ are closer to each other, being that the only method the proposal does not improve.

6. Conclusions

In this paper, a combination of a Genetic Algorithm (GA) and a Cross Entropy (CE) method for continuous optimization was presented.

The development of this algorithm have been motivated by the aim of combine the exploration ability of a GA with the exploitation ability of the CE in order to solve the lacks both algorithms have. Besides, it is the first time GA is combined with CE for this kind of problems.

The method divides the population into two sub-populations. In one of them, a GA is applied while in the other one, the CE method is executed. After that, both resulting sub-populations are joined, and they replace the current population. A total of 24 functions have been obtained from Black-Box Optimization Benchmarking (BBOB), of different types: from separable functions to multi-modal ones.

First, a study to determine the population size was made. With the conclusions obtained, an experiment to determine the value of GA percentage in order to determine the amount of GA and CE individuals in each sub-population was made. In addition, for the CE part, the amount of individuals necessary to update the means and standard deviations had to be defined. After studying the different combinations of these parameters, it resulted that good performance was obtained for low values of these parameters. These values were then employed in a comparison with several standard algorithms in the literature. The proposal obtained at least one of the two best values in 62 of 96 cases of study.

These conclusions were supported by statistical tests such as Friedman's, where the proposal reaches the best ranking in larger dimensions. Also, the tests of Holm and Finner have been used in order to see if the results of the experimentation are significantly different from those obtained by the literature algorithms.

As experimentation shows, the number of best values increased with larger dimensions. One of the limitations found during the experiments was the poor performance in low dimensions when the proposal is compared with algorithms as DE in the same dimensions. Since the proposal is a population-based method, when number of individuals is low, diversity can be affected along the generations.

In future research, real-world functions can be used in order to determine whether the proposal provides good results in these cases with the parameters obtained in this study. In addition, the use of more dimensions, up to 40, or down to 5, can be studied. Another suggestion can be the adaptation of the proposal to multi-objective functions, using each sub-population for one objective. Improve the performance in low dimensions, and a study about the diversity in both sub-populations can be taken into account for future works. Additionally, the adaptive selection of the parameters for each part of the algorithm by means of co-evolution, that is, evolving the parameters together as a part of the chromosome, can be interesting research line to explore. Finally, the comparative with other methods such as CMA, NSGA-II or memetic algorithms can be taken into account for future works.

Appendix A. Benchmark Function Formulas

Each table shows each function type, the functions of that type, and its mathematical form in order to help with its understanding. About variables $z_i, s_i, \mathbf{z}$ and others which appear in the most of the functions, they have its value in the documentation that the reader can find in the link below [3].

Name	Formula
Sphere	$F_1(x) = x - x^{opt} + F_{opt}$
Ellipsoidal	$F_2(x) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + F_{opt}$
Rastrigin	$F_3(x) = 10\left(D - \sum_{i=1}^{D} \cos(2\pi z_i)\right) + \|\|\mathbf{z}\|\|^2 + F_{opt}$
Bche-Rastrigin	$F_4(x) = 10\left(D - \sum i = 1^D \cos 2\pi z_i\right) + \sum i = 1^D + 100 f_{pen} x + F_{opt}$
Linear Slope	$F_5(x) = \sum i = 1^D 5\|s_i\| - s_i z_i + F_{opt}$

Table A.11: Separable functions: Names and formulas.

[3] http://coco.gforge.inria.fr/lib/exe/fetch.php?media=download3.6:bbobdocfunctions.pdf

Name	Formula
Attractive Sector	$F_6(x) = T_{osz}\left(\sum i = 1^D (s_i z_i)^2\right)^{0.9} + F_{opt}$
Step Ellipsoidal	$F_7(x) = 0.1max\left(\frac{\lvert z_1\rvert}{10^4}, \sum i = 1^D 10^{2\frac{i-1}{D-1}} z_i^2\right) + f_{pen}(x) + F_{opt}$
Rosenbrock , original	$F_8(x) = \sum_{i=1}^{D-1}\left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right) + F_{opt}$
Rosenbrock , rotated	$F_9(x) = \sum_{i=1}^{D-1}\left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right) + F_{opt}$

Table A.12: Low or Moderate conditioning functions: Names and formulas.

Name	Formula
Ellipsoidal	$F_{10}(x) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D?1}} z_i^2 + F_{opt}$
Discus	$F_{11}(x) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + F_{opt}$
Bent Cigar	$F_{12}(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + F_{opt}$
Sharp Ridge	$F_{13}(x) = z_1^2 + 100\sqrt{\sum_{i=2}^{D} z_i^2} + F_{opt}$
Different Powers	$F_{14}(x) = \sqrt{\sum_{i=1}^{D} \lvert z_i\rvert^{2+4\frac{i-1}{D-1}}} + F_{opt}$

Table A.13: High conditioning and unimodal functions: Names and formulas.

Name	Formula
Rastrigin	$F_{15}(x) = 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + \lvert\lvert\mathbf{z}\rvert\rvert^2 + F_{opt}$
Weirstrass	$F_{16}(x) = 10\left(\frac{1}{D}\sum_{i=1}^{D}\sum_{k=0}^{11}\frac{1}{2}^k \cos(2\pi 3^k(z_i + \frac{1}{2})) - f_0\right)^3 + \frac{10}{D} f_{pen}(x) + F_{opt}$
Schaffers F7	$F_{17}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}\sin^2(50 s_i^{1/5})\right)^2 + 10 f_{pen}(x) + F_{opt}$
Schaffers F7, moderately ill-conditioned	$F_{18}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}\sin^2(50 s_i^{1/5})\right)^2 + 10 f_{pen}(x) + F_{opt}$
Composite Griewank-Rosenbrock	$F_{19}(x) = \frac{10}{D-1}\sum_{i=1}^{D-1}\left(\frac{s_i}{4000} - \cos s_i\right) + 10 + F_{opt}$

Table A.14: Multi-modal with adequate global structure functions: Names and formulas.

Name	Formula
Schwefel	$F_{20}(x) = -\frac{1}{D}\sum_{i=1}^{D} z_i \sin(\sqrt{\lvert z_i \rvert}) + 4.189828872724339 + 100 f_{pen}(\frac{\mathbf{z}}{100}) + F_{opt}$
Gallagher's Gaussian 101-me Peaks	$F_{21}(x) = T_{osz}\left(10 - \max_{i=1}^{101} w_i \exp\left(-\frac{1}{2D}(x-y_i)^T R^T C_i R(x-y_i)\right)\right)^2 + f_{pen}(x) + F_{opt}$
Gallagher's Gaussian 21-hi Peaks	$F_{22}(x) = T_{osz}\left(10 - \max_{i=1}^{21} w_i \exp\left(-\frac{1}{2D}(x-y_i)^T R^T C_i R(x-y_i)\right)\right)^2 + f_{pen}(x) + F_{opt}$
Katsuura	$F_{23}(x) = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{\lvert 2^j z_i - [2^j z_i]\rvert}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{pen}(x)$
Lunacek bi-Rastrigin	$F_{24}(x) = \min\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + 10^4 + f_{pen}(x)$

Table A.15: Multi-modal with weak global structure functions: Names and formulas.

Acknowledgments

This research work is supported by TIMON Project (Enhanced real time services for an optimized multimodal mobility relying on cooperative networks and open data). This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 636220.

References

Abadlia, H., Smairi, N., & Zidi, K. (2014). A study of the efficiency of hybridized approaches based on particle swarm optimization technique. (pp. 190–199). volume 1.

Abd-El-Wahed, W., Mousa, A., & El-Shorbagy, M. (2011). Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. *Journal of Computational and Applied Mathematics*, *235*, 1446–1453.

Alba, E. (2005). *Parallel metaheuristics: a new class of algorithms* volume 47. John Wiley & Sons.

Alba, E., & Dorronsoro, B. (2009). *Cellular genetic algorithms* volume 42. Springer-Verlag.

Asafuddoula, M., Ray, T., & Sarker, R. (2011). An adaptive differential evolution algorithm and its performance on real world optimization problems. (pp. 1057–1062).

Auger, A., Brockhoff, D., & Hansen, N. (2013). Benchmarking the local metamodel cma-es on the noiseless bbob'2013 test bed. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (pp. 1225–1232). ACM.

Bäck, T., & Schwefel, H. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, *1*, 1–23.

Blum, C., Puchinger, J., Raidl, G., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing Journal*, *11*, 4135–4151.

Bolaji, A., Khader, A., Al-Betar, M., & Awadallah, M. (2015). A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems. *Journal of Intelligent Systems*, *24*, 37–54.

Busoniu, L., Ernst, D., De Schutter, B., & Babuska, R. (2011). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *41*, 196–209.

Caballero, R., Hernndez-Daz, A., Laguna, M., & Molina, J. (2015). Cross entropy for multiobjective combinatorial optimization problems with linear relaxations. *European Journal of Operational Research*, *243*, 362–368.

Cai, W., & Ma, L. (2010). Applications of critical temperature in minimizing functions of continuous variables with simulated annealing algorithm. *Computer Physics Communications*, *181*, 11–16.

Chen, J., Xin, B., Peng, Z., & Pan, F. (2009). Statistical learning makes the hybridization of particle swarm and differential evolution more efficienta novel hybrid optimizer. *Science in China Series F: Information Sciences*, *52*, 1278–1282.

Ciornei, I., & Kyriakides, E. (2012). Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *42*, 234–245.

Cotta-Porras, C. (1998). Study of hybridization techniques and their application to the design of evolutionary algorithms. *AI Communications*, *11*, 223–224.

Deb, K., Anand, A., & Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, *10*, 371–395.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*, 3–18.

Dorigo, M., & Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*, 53–66.

Duan, H., Luo, Q., Shi, Y., & Ma, G. (2013). Hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration. *IEEE Computational Intelligence Magazine*, *8*, 16–27.

Elsheikh, A. (2014). Derivative-based hybrid heuristics for continuous-time simulation optimization. *Simulation Modelling Practice and Theory*, *46*, 164–175.

Finner, H. (1993). On a monotonicity problem in step-down multiple test procedures. *Journal of the American Statistical Association*, *88*, 920–923.

Fujikawa, M., & Takashi, M. (2005). Modified intelligent hybrid technique reducing experimental error over the entire target area. *Experimental Mechanics*, *45*, 541–549.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, *1*, 69–93.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*, 159–195.

Hernández, S., Leguizamón, G., & Mezura-Montes, E. (2013). Hybridization of differential evolution using hill climbing to solve constrained optimization problems. *Revista Iberoamericana de Inteligencia Artificial*, *16*, 3–15.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, *12*, 265–319.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Univ. of Michigan Press.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, (pp. 65–70).

Holtschulte, N. J., & Moses, M. (2013). Benchmarking cellular genetic algorithms on the BBOB noiseless testbed. In *Proceedings of the Companion Publication of the 2013 Conference on Genetic and Evolutionary Computation* (pp. 1201–1208). ACM.

Hutter, F., Hoos, H., & Leyton-Brown, K. (2013). An evaluation of sequential model-based optimization for expensive blackbox functions. In *Proceedings of the Companion Publication of the 2013 Conference on Genetic and Evolutionary Computation* (pp. 1209–1216). ACM.

Jacobson, S. H. e. a. (2004). Global optimization performance measures for generalized hill climbing algorithms. *Journal of Global Optimization*, *29*, 173–190.

Jourdan, L., Basseur, M., & Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, *199*, 620–629.

Karakatic, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, *27*, 519–532.

Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning* (pp. 760–766). Springer-Verlag.

Kıran, M. S., & Gündüz, M. (2013). A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. *Applied Soft Computing*, *13*, 2188–2203.

Kroese, D., Porotsky, S., & Rubinstein, R. (2006). The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, *8*, 383–407.

Kumar, D., & Beniwal, S. (2013). Genetic algorithm and programming based classification: A survey. *Journal of Theoretical and Applied Information Technology*, *54*, 48–58.

Lim, T. (2014). Structured population genetic algorithms: A literature survey. *Artificial Intelligence Review*, *41*, 385–399.

Lopez-Garcia, P., Onieva, E., Osaba, E., Masegosa, A. D., & Perallos, A. (). A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy, .

Lopez-Garcia, P., Onieva, E., Osaba, E., Masegosa, A. D., & Perallos, A. (2015). Hybridizing genetic algorithm with cross entropy for solving continuous functions. In *Proceedings of the Companion Publication of the 2015 Conference on Genetic and Evolutionary Computation* GECCO Companion '15 (pp. 763–764).

Mandal, A., Das, A., Mukherjee, P., Das, S., & Suganthan, P. (2011). Modified differential evolution with local search algorithm for real world optimization. (pp. 1565–1572).

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, *33*, 61–106.

Olama, M., Ma, X., Killough, S., Kuruganti, T., Smith, S., & Djouadi, S. (2015). Analysis, optimization, and implementation of a hybrid DS/FFH spread-spectrum technique for smart grid communications. *Eurasip Journal on Advances in Signal Processing*, *2015*.

Onieva, E., Naranjo, J., Milanés, V., Alonso, J., García, R., & Pérez, J. (2011). Automatic lateral control for unmanned vehicles via genetic algorithms. *Applied Soft Computing*, *11*, 1303–1309.

Osaba, E., Diaz, F., & Onieva, E. (2014). Golden ball: A novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. *Applied Intelligence*, *41*, 145–166.

Osaba, E., Onieva, E., Carballedo, R., Diaz, F., Perallos, A., & Zhang, X. (2013). A multi-crossover and adaptive island based population algorithm for solving routing problems. *Journal of Zhejiang University SciencE C*, *14*, 815–821.

Osman, I., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, *63*, 513–623.

Pošík, P., & Klemš, V. (2012). Benchmarking the differential evolution with adaptive encoding on noiseless functions. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation* (pp. 189–196). ACM.

Purwar, A., & Singh, S. (2015). Hybrid prediction model with missing value imputation for medical data. *Expert Systems with Applications*, *42*, 5621–5631.

Qiao, J., Yang, N., & Gao, J. (2011). Two-stage fuzzy logic controller for signalized intersection. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *41*, 178–184.

Rahman, D., Viana, A., & Pedroso, J. (2014). Metaheuristic search based methods for unit commitment. *International Journal of Electrical Power and Energy Systems*, *59*, 14–22.

Raidl, G. (2006). A unified view on hybrid metaheuristics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4030 LNCS*, 1–12.

Reale, T., & OConnor, A. (2011). Cross-entropy as an optimization method for bridge condition transition probability determination. *Journal of Transportation Engineering*, *138*, 741–750.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, *1*, 127–190.

Sawyerr, B. A., Adewumi, A. O., & Ali, M. (2013). Benchmarking projection-based real coded genetic algorithm on BBOB-2013 noiseless function testbed. In *Proceedings of the Companion Publication of the 2013 Conference on Genetic and Evolutionary Computation* (pp. 1193–1200).

Segura, C., Coello, C. A. C., & Hernández-Díaz, A. G. (2015). Improving the vector generation strategy of differential evolution for large-scale optimization. *Information Sciences*, *323*, 106–129.

Shim, V., Tan, K., & Tang, H. (2015). Adaptive memetic computing for evolutionary multiobjective optimization. *IEEE Transactions on Cybernetics*, *45*, 610–621.

Sierra, L.-M., Cobos, C., & Corrales, J.-C. (2014). Continuous optimization based on a hybridization of differential evolution with K-means. *Lecture Notes in Computer Science*, *8864*, 381–392.

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of heuristics*, *8*, 541–564.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*.

Thakur, M. (2014). A new genetic algorithm for global optimization of multimodal continuous functions. *Journal of Computational Science*, *5*, 298–311.

Topcuoglu, H. R., Demiroz, B., & Kandemir, M. (2007). Solving the register allocation problem for embedded systems using a hybrid evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, *11*, 620–634.

Tran, T.-D., Brockhoff, D., & Derbel, B. (2013). Multiobjectivization with nsga-ii on the noiseless bbob testbed. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (pp. 1217–1224). ACM.

Van Laarhoven, P. J., & Aarts, E. H. (1987). *Simulated Annealing*. Springer-Verlag.

Voglis, C. (2013). Adapt-mempsode: a memetic algorithm with adaptive selection of local searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (pp. 1137–1144). ACM.

Wang, N., Zhang, X., & Yang, Y. (2013). A hybrid genetic algorithm for constrained multi-objective optimization under uncertainty and target matching problems. *Applied Soft Computing Journal*, *13*, 3636–3645.

Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., & Tian, Q. (2011). Self-adaptive learning based particle swarm optimization. *Information Sciences*, *181*, 4515–4538.

Wang, Z., Wong, Y., & Rahman, M. (2004). Optimisation of multi-pass milling using genetic algorithm and genetic simulated annealing. *The International Journal of Advanced Manufacturing Technology*, *24*, 727–732.

Zhao, W.-Z., Wang, C.-Y., Yu, L.-Y., & Chen, T. (2013). Performance optimization of electric power steering based on multi-objective genetic algorithm. *Journal of Central South University*, *20*, 98–104.